A sparse recovery view of sentence embeddings, bag of n-grams, and LSTMs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Low-dimensional vector embeddings capturing the "meaning" of text are popular, often computed using LSTMs as well as some simpler techniques.

Here such text embeddings are related to a more basic method for representing text, Bag-of–n-Gram (BonG) vectors. We suggest that much of the power of low-dimensional embeddings can be derived from compressed versions of BonG vectors. We view simple unsupervised distributional word embeddings based upon simple tensor products —similar to those reported in recent papers— as low-dimensional linear transformations of BonG vectors. Using standard theory from sparse recovery/compressed sensing we can prove that the original BonG information is retained in these simple embeddings.

This understanding leads to a new theoretical result about LSTMs: low-dimensional embeddings derived from a low-memory LSTM are provably at least as powerful on classification tasks as any linear classifier (SVM or regression) trained over the full BonG vectors. Note that the latter is often considered a fairly tough benchmark for embeddings in practice.

Experimental results are presented supporting the above theoretical result. Another intriguing empirical finding is that the standard word embeddings (eg GloVe or word2vec), when viewed as a matrix, allow a more powerful compression/encoding for BonG vectors than would be suggested by standard compressed sensing.

1 Introduction

Vector embeddings for text have become very popular in many applications. A piece of text –sentence or paragraph— is passed through some variant of an LSTM (recurrent neural nets with Long Short Term Memory) (Hochreiter et. al. 1997)[9]. The LSTM sweeps once or twice through the text using limited memory to process it and finally output a single vector with moderate dimensionality (a few hundred to a few thousand) that seems to capture its essential structure. For example, the vector representation can be used to judge similarity of two pieces of text, or as a sentence featurization in downstream classification tasks. A cottage industry of ideas has grown around showing that the performance of the best embeddings produced by LSTMs can be matched by much simpler and transparent text embeddings, as surveyed below.

Vector representations of concepts—known also as *distributed representations*— have a very long history in connectionist approaches, since they decay gracefully with noise and allow fast processing using simple and distributed computational elements. Hinton (1990)[8] lays out design criteria for them. Plate (1995)[18] provided a notable solution to Hinton's problem, the *Holographic Distributed Representation*. It represents structured objects using convolutional product of vectors and achieves fast implementation times using the fast Fourier transform. Plate suggested applications of his ideas to

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

text, since "structure" in text can be quantified using parse trees and other syntactic and grammatical structures.

The goal of the current paper is to relate recent sentence embeddings to these older notions. Simplified notions of Plate's ideas can be applied to the n-gram representation of text (Bag of n-grams or BonG). Applying them to unigram representation yields the simple sentence embeddings of (Wieting et. al. 2016, Arora et. al. 2017)[1, 23]. Applying them to unigram plus bigrams gives even stronger embeddings that appear to be new and get better performance in downstream tasks. The advantage of this approach is that the performance of the embeddings can be rigorously understood using standard notions in the theory of sparse recovery.

We can prove (Section 4) that the performance of our embeddings on downstream classification tasks approaches that of the best bigram-based SVMs, which is considered a benchmark —albeit a tough one— for good text embeddings. (Note that our text embeddings are constructed using unlabeled data, i.e., are unsupervised.) Section 5 presents experimental results supporting this theory. Our embedding can also be computed by a simple LSTM as well, which needs to be initialized with some vector embeddings for individual words e.g., word2vec or GloVe. The LSTM's internal memory is very tightly limited, except for the bank of pre-trained word embeddings. (By slightly relaxing the limit on internal memory, one can even use random vectors instead of word embeddings.) Thus our results imply that the theoretical performance of LSTMs, even one with a strong upper limit on internal memory, is at least as good as that of an SVM operating on the full n-gram representation. This result is novel in deep learning theory.

While the above result can be proved using random vectors as word embeddings, we find that pretrained embeddings work much better at preserving BonG information, much better than the theoretical limits usually discussed in theory of compressed sensing. We provide some intuition and evidence supporting this intuition later in section 4.

2 The embeddings

The Bag-of-Words (BoW) representation for a sentence is a vector of length V (vocabulary size) where the i^{th} entry is the number of occurrences of the word i in the sentence. To capture word order information, BoW representations can be extended to count occurrences of all n-grams — contiguous sequence of n words — in addition to unigrams (Harris 1954)[6]. We call this representation the Bag-of-n-Grams (BonG) representation of the sentence. The resulting representation is a sparse vector of dimension $\mathcal{O}(V^n)$. However, in practice the number of n-grams is of order $\mathcal{O}(V)$.

Our sentence embeddings use a column vector $v_w \in \mathbb{R}^d$ for each word w in the vocabulary ("word embedding"). Given a sentence with the word sequence $w_1, w_2, \ldots, w_k$, the *simple unigram embedding* is $\sum_i v_{w_i}$. We define the *simple bigram embedding* as follows: $\sum_i v_{w_i} v_{w_{i+1}}^T$. which is a $d \times d$ matrix. The simple sentence embedding is the concatenation of the unigram and bigram embeddings. This approach generalizes to n-gram embeddings via tensor products: bag-of-n-grams $(\langle w_1, w_2, \cdots, w_k \rangle) = \sum_{i=1}^{k} \bigotimes_{j=i}^{i+n-1} v_{w_j}$.

Note that if the dimension d equals the vocabulary size V and if v_w is the one-hot encoding for the word w, then the simple unigram embedding is exactly the BoW representation, and the simple bigram embedding is exactly the bigram vector. So this sentence embedding is a generalization of the BonG representation. We are interested in the case $d \ll V$, when the embedding must lose information in general.

The starting point of our work is the observation that the simple unigram embedding is a linear transformation of the BoW vector: If A is the $d \times V$ matrix whose w column is v_w, and x is the BoW representation of the sentence, then the embedding is Ax. (Likewise for the bigram embedding.) Our results leverage the linear algebraic properties of A.

We also define weighted versions of these embeddings. In training bigram models, especially with limited labeled data, it can be useful to assign a nonnegative weight to each word. If $\beta(w)$ is the weight assigned to word w, and $\gamma(w, w')$ is the weight assigned to the bigram w, w', then the weighted unigram embedding is $\sum_i \beta(w_i) v_{w_i}$ and the weighted bigram embedding is $\sum_{ij} \gamma(w_i, w_j) v_{w_i} v_{w_j}^T$.

Two specific choices of word embeddings will be of interest: (i) each v_w is a random d-dimensional unit vector (alternatively, vectors with i.i.d. coordinates from $\mathcal{N}(0, 1/d)$), (ii) v_w is computed using a

standard embedding technique like GloVe (Pennington et. al. 2014)[17] or word2vec (Mikolov et. al. 2013)[14] trained on a standard text corpus.

The bigram embedding squares the dimensionality, so we also define a more efficient *dimension reduced* version of the bigram embedding. We can implement dimension reduction by selecting a random sequence of coordinate pairs $(i_1, j_1), (i_2, j_2), \ldots, \in \{1, 2, \ldots, d\}$ and reporting only the corresponding entries in the bigram embedding. More complicated dimension reduction approaches using convolution appear in Plate's work and a few others are described in the empirical results section.

Clearly, each of these embeddings is computable with a very simple LSTM that has access to the word embeddings and unigram and bigram weights. For example, the LSTM for the simplest embedding makes one pass through the text, maintaining the embedding of the text seen so far. If the ith word is w_i, it adds the vector v_{w_i} to the unigram embedding, as well as the vector $v_{w_{i-1}} v_{w_i}^T$ to the bigram embedding. The LSTM for the weighted embeddings is analogous. The next theorem shows the power of this LSTM.

Theorem 2.1 (LSTMs do at least as well as linear classification over BonGs)**.** *When the word embeddings are random Gaussian embeddings with dimension $d \geq \mathcal{O}\left(k \log \frac{V}{k}\right)$, the above simple LSTM with $\mathcal{O}(d)$ memory can have performance on classification tasks at least as good as the standard SVM/regression given the full Bag-of-n-Grams vector.*

3 Related Work

The simple unigram embedding (known as *average of word vectors*) is ubiquitous in natural language processing and its power was explored in (Wieting et. al. 2016)[23]. A weighted unigram embedding appears in (Arora et. al. 2017)[1], which gives a justification for the weights using a generative model of text. The idea of using vector outer product to represent bigrams is also old, but surprisingly, our precise embedding (concatenation of unigram and bigram) does not appear to have been studied as a sentence embedding. The contribution of our paper is to give a theoretical justification of this strange-looking embedding and to demonstrate its utility.

Using vector outerproducts to represent bigrams (and more generally, tensor products to represent n-grams) has the drawback of blowup in dimension. To combat this, Plate (1995)[18] introduced convolutional embeddings. These representations may be viewed as a sketch of the outer product matrix, with the nice property that they maintain the dimension of the original embeddings. Plate used the fast Fourier transform in his algorithm. Since we use compressed sensing ideas, we can use a random subsample of the tensor product.

Other approaches for compressing representations of text exist. Paskov et. al. (2013)[15] introduce text compression methods from classic information theory to the task of feature representation, and with compressed representations are able to replicate the performance of their uncompressed features. However, these features do not approach the performance of BonG features.

We can also think of LSTMs as encoding a sequence representation for its hidden state to hold — if the number of parameters in the LSTM unit is small, then we can also think of this as a low-dimensional embedding. Typically the LSTM parameters are trained using backpropagation but the simple LSTM of Theorem 1 needs no training.

Compressed sensing recap. Our results use the basic idea of compressed sensing and the result of Calderbank et al. [2] demonstrating that it is possible to learn SVMs with low-dimensional embeddings of the data space.

Definition 3.1 (Restricted Isometry Property)**.** *A matrix $A \in \mathbb{R}^{m \times n}$ is (k, ϵ)-RIP if for all k-sparse x*

$$(1 - \epsilon)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \epsilon)\|x\|_2^2 \tag{1}$$

Simple manipulation gives the following.

Lemma 3.1 (Inner product bound)**.** *For k-sparse x, y, with A $(2k, \epsilon)$-RIP and $\|x\|_2, \|y\|_2 \leq R$,*

$$(1 + \epsilon)x^T y - 2R^2\epsilon \leq (Ax)^T(Ay) \leq (1 - \epsilon)x^T y + 2R^2\epsilon \tag{2}$$

Since the SVM classifier depends upon pairwise inner products of datapoints, Calderbank et al. [2] use the above facts to show that training an SVM on the compressed vectors Ax is almost as good as training on the original vector x. This uses the observation that the RIP property hold for not just sparse vectors but also all vectors that are in the convex hull of the sparse vectors. The SVM classifier over training data $S = \{(x_i, y_i)\}_{i=1}^M$ can be written as $\hat{w}_S = \sum_{i=1}^M \alpha_i y_i x_i$. Then, applying the RIP properties from before to this equation yields the bound $(Aw)^T(Aw) \leq w^T w + 3C^2R^2\epsilon$, where $\sum_{i=1}^M \alpha_i \leq C$ is the weight parameter of the C-SVM. This inequality is then used to directly show the gap between the performance of the SVM learned in the data domain (high-dimensional) and the SVM learned in the measurement domain projection via A (low-dimensional).

4 Provable properties of sentence embeddings

The main contribution of the current work is to justify these simple embeddings using theory of sparse recovery (also called compressed sensing). By contrast, a recent work (Arora et. al. 2017)[1] had tried to justify the simple weighted unigram sentence embedding using a generative model of text. That justification does not appear to extend to our bigram embedding. Throughout, k, d, V have the meanings described in Section 2.

4.1 Connecting sparse bag-of-n-grams and our embeddings

As a warm-up, we note that if the dimension of the embedding is allowed to be as large as V then our embeddings precisely preserve the unigram and bigram information (use 1-hot binary vectors as word embeddings). Here we are interested in dimension $d \ll V$, so instead of 1-hot embeddings one can approximately preserve the unigram and bigram information if the low-dimensional embeddings are roughly orthogonal.

For simplicity of notation we give the formal analysis for the unigram embeddings, but the proof for bigram embeddings is the same. Let $x \in \mathbb{R}^V$ denote the unigram representation, meaning the coordinate for word w is the number of its occurences in the sentence. Then our embedding can be seen as matrix-vector product Ax where A is a matrix whose w'th column is the embedding (possibly scaled by the word's weighting) for word w. In what follows, it is important that $k \ll V$, meaning x is a sparse vector.

To show that the embedding Ax is as useful as x for downstream classification tasks using an SVM, we adapt results of (Calderbank et al. 2009)[2] on doing classification with compressed sensing. They use standard analysis of SVMs (Shalev-Schwartz, Ben David) which implies that the quality of the training is not greatly affected if the transformation $x \to Ax$ approximately preserves inner product.

$$\langle x, x' \rangle - \delta \leq \langle Ax, Ax' \rangle \leq \langle x, x' \rangle + \delta \tag{3}$$

This cannot be true in general for all x but it can be true for x that are k-sparse for small k. The precise dependence of k on d, V is given by theory of compressed sensing.

We can also extend the theorem from SVM classifiers to all classifiers trained using ℓ_2 regularization with Lipschitz convex loss.

Theorem 4.1. *(General compressed learning.) Suppose that we are trying to train a classifier with an C-Lipschitz, differentiable convex loss function with ℓ_2 regularization. Then if an embedding matrix A is $(2k, \epsilon)$-RIP, and we train on training set $S = \{(x_1, y_1), \cdots, (x_M, y_M)\}$, the loss of the classifier learned in the low-dimensional regime $\{(Ax_1, y_1), \cdots, (Ax_M, y_M)\}$ $L_D(A\hat{w}_S)$ is related to the loss in the data domain by*

$$L_D(A\hat{w}_S) \leq L_D(\hat{w}_S) + \mathcal{O}(CR^2\epsilon) \tag{4}$$

Proof. We want to minimize over w the loss function $L(w) = \frac{1}{2}\|w\|_2^2 + \frac{1}{M}\sum_{i=1}^M l(w^T x_i, y_i)$, where we assume l is both convex and C-Lipschitz. Since l is convex and differentiable, setting the derivative w.r.t. w to 0 will give us the optimal w. By chain rule, we get $w + \frac{1}{M}\sum_{i=1}^M \frac{\partial l(w^T x_i, y_i)}{\partial (w^T x_i)} x_i = 0$. Thus $w = \frac{1}{M}\sum_{i=1}^M -\frac{\partial l(w^T x_i, y_i)}{\partial (w^T x_i)} x_i = \sum_{i=1}^M \gamma_i x_i$. Using the Lipchitz assumption, we get $\sum_{i=1}^M \gamma_i \leq C$ and we can plug this expression to Theorem 4.4 and Lemma 5.1 in (Calderbank et al. 2009) to achieve the desired result. □

4.2 Proving Theorem 2.1

Using the Calderbank et. al. analysis, we can now prove Theorem 2.1.

Proof. We first note that a random matrix $A \in \mathbb{R}^{d \times V}$ with i.i.d. $\mathcal{N}(0, \frac{1}{d})$ entries satisfies the RIP constant $\delta_k < \delta$ with high probability as long as $d \gtrsim \frac{1}{\delta^2} k \log \frac{V}{k}$, where δ_k is the RIP coefficient for sparsity k (Hastie et. al. 2015) [7]. When the word embeddings are random vectors, the analysis of Calderbank et. al. (2009) [2] can be applied as is since the matrix A has the $(2k, \epsilon)$-RIP property, which implies (3) for an appropriate δ. Therefore, d-dimensional representations are able to achieve the same performance as their high-dimensional versions. By describing the embeddings as a sum of unigram embeddings (v_{w_i}) concatenated with a sum of bigram embeddings $(v_{w_i} v_{w_j}^T)$ in terms of the parameters of the LSTM, we have therefore shown that the model class of LSTMs which produce sentence embeddings is strictly larger than the class of bag-of-n-gram embeddings. Thus, an LSTM exists which performs better than SVM over a bag-of-n-grams representation. □

Random embeddings also have an explicit guarantee on their ability to reconstruct bag-of-n-grams representations. Compressed sensing allows the stronger result that x can be recovered from Ax for any k-sparse x (Hastie et. al. 2015) [7].

Theorem 4.2 (Recovery)**.** *Suppose that a bag-of-words representation x is k-sparse. If A's columns are i.i.d. d-dimensional Gaussian vectors with termwise variance $1/d$, and $d \gtrsim \frac{2k}{\delta_{2k}^2} \log \frac{V}{2k}$, then A is $(2k, \delta_{2k})$-RIP w.h.p. Then if $\delta_{2k} < \sqrt{2} - 1$, ℓ_1-minimization algorithm for sparse recovery (Hastie et. al. 2015)[7] succeeds in recovering the sparse vector exactly in polynomial time. Therefore the original sentence can be recovered via an ℓ_1 convex program from its random unigram embedding in A. (A similar statement is true for the sentence embedding with bigrams.)*

4.3 Using pretrained word embeddings

In practice, trained word embeddings are a better idea than random vectors because they capture word similarity. If *happy* and *delighted* have similar word vectors, then their word embeddings will contribute similarly to the overall sentence embedding. This intuition is well-known but here we are exploring their linear algebraic properties, since as pointed out, our sentence embedding is merely a matrix transformation A applied to the BonG vector where the columns of A are word embeddings.

The RIP property does not hold for such an A even for $k = 2$ once we replace random word embedding with any standard word embedding such as word2vec or GloVe, since there are now pairs of words (e.g., synonyms) could be represented by unit vectors that have high inner product, or even inner product 1 (i.e. the two vectors are the same). Thus no theorem analogous to Theorem 4.2 is possible now.

However, the RIP property is relevant only if we are interested in applying compressed sensing for *all* k-sparse vectors. In practice the k-sparse vectors of interest correspond to an actual piece of text, whose BonG vector is not an arbitrary k-sparse vector. How well does compressed sensing work for such vectors? If the compressed sensing properties hold for just these vectors, then the result of Calderbank et al. still goes through. (Since it only uses inner products among points in the dataset.)

In Section 5 we report empirical findings that the sparse recovery procedure of Theorem 4.2 does work *much better* for such BonG vectors than compressed sensing theory would suggest. In other words, these trained vectors are better for compressed sensing than random vectors, which are theoretically near-optimal for sensing. Theoretically explaining this fact about trained vectors is left for future work. But it does lend credence to our thesis that sparse recovery plays an important role in explaining the power of text embeddings.

5 Empirical Findings

Experiments.

1. We tested sparse recovery of the original BoW representation of a sentence from a sentence embedding using Lasso regression [7]. We tried recovering real sentences from datasets and

also random k-sparse BoW vectors. Figures 5 and 5 suggest that word embeddings do much better at recovering BonGs for actual sentences from the datasets.

2. We compare the performance of unsupervised BonG models with unsupervised word embedding models on two standard binary classification tasks - the Stanford Sentiment Treebank (SST) [21] and the Internet Movie Database review corpus (IMDB) [12]. For both we consider a logistic regression classifier with l^2-regularization, with the regularization coefficient determined by cross-validation on the training data.

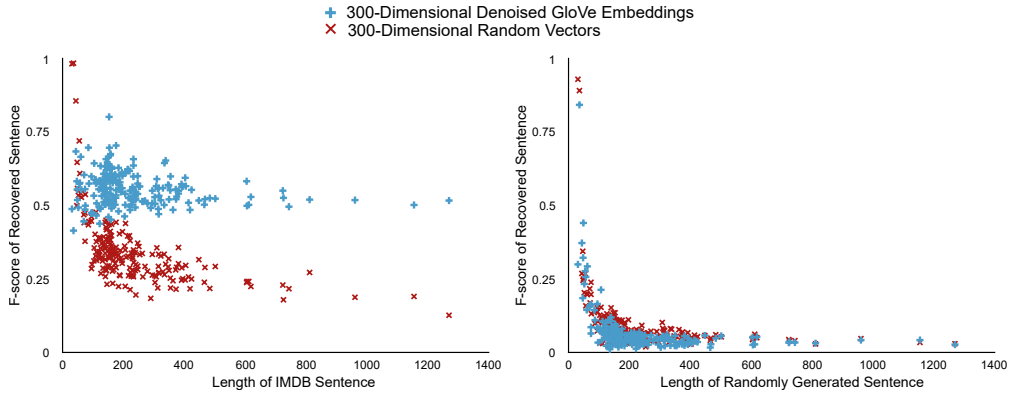

Figure 1: Comparison of BoW sentence recovery using LASSO on real sentences from the IMDB dataset (left) and fake sentences generated by picking words at random (right). Trained word embeddings (GloVe) are able to reconstruct the original BoW vector much better using the real sentences.

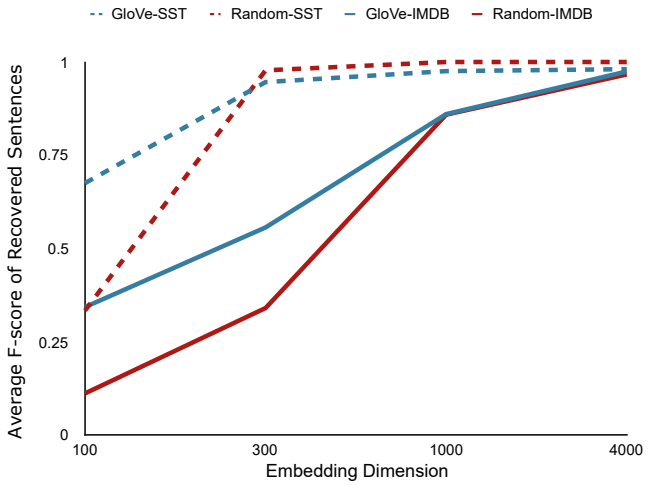

Figure 2: F_1-score compared to the dimension when recovering SST and IMDB sentences using denoised and projected GloVe embeddings vs. random vectors. GloVe embeddings (in blue) require a much lower dimension to reconstruct the original BoW vector. Note that SST has much shorter sentences on average than IMDB (13 vs. 270) and is thus much easier for both embeddings to recover.

Word embeddings. We perform experiments primarily using 100, 300 and 1000 dimensional word embeddings trained on Amazon Product Data Corpus [13] using the GloVe [17] objective. These embeddings seems to perform consistently better on sentiment classification tasks than GloVe vectors trained on the CommonCrawl [4] corpus and this dataset was also used by Radford et. al. (2017) [19] to train the LSTM that found the sentiment neuron. We will call these embeddings Amazon GloVe embeddings.

Sketching the outer product. We consider various sketches of outer product (OP) for bigram embeddings.

1. **Concat** is the concatenation of the constituent unigrams in the bigrams.
2. **EMul** is the element-wise multiplication of the constituent unigrams in the bigram, where the second unigram is multiplied by a random matrix. So $v_{a,b} = v_a \odot Mv_b$ where M is matrix whose entries are normally distributed and independent. The inner products between a pair of these bigram embeddings is an unbiased estimator of the inner product between the outer product representations of the bigram embeddings, given by $\langle v_a v_b^T, v_x v_y^T \rangle = \text{Tr}\left(v_b^T v_a v_x v_y^T\right) = (v_a^T v_x)(v_b^T v_y)$.
3. **OP+JL** uses the Johnson-Lindenstrauss random dimension reduction approach to arrive at a d (or $10d$)-dimensional representation of the outer product matrix such that inner products are maintained up to an additive error.
4. **OP+sample** randomly samples d (or $10d$) coordinates from the d^2 coordinates outer product matrix to arrive at a representation, where d is the dimension of the original embeddings.

SIF-Weighted embeddings. The representations used in Arora et. al. 2017 [1] apply unsupervised weights known as **smooth inverse frequency** (SIF) weights to the word embeddings before summing the embeddings together. A SIF weight for word w is defined by $\frac{\alpha}{\alpha+f_w}$, where f_w is the frequency of the word in a corpus. These embeddings down-weight the most frequent words.

Denoised embeddings. The theory in the previous sections suggest that a RIP property is important for recovering BonG representations from low-dimensional random embeddings, and is also important for achieving classification performance which matches BonGs. Therefore, for the standard word embeddings which are **not** random and do not *a priori* have RIP-like properties, we would like to encourage them to have properties closer to RIP while still preserving similarity information.

We achieve this intuition by **denoising** the inner product matrix of the embeddings: We set all inner products below a certain threshold (0.3 in our experiments) to zero and leave the remaining entries unchanged. We then factor this "denoised" similarity matrix using SVD to get embeddings in the same (or possibly higher) dimensional space. Thus we preserve the high inner products and send dissimilar words further apart. In practice we observe that these denoised embeddings consistently perform better than the original embeddings. Average improvement on 300 dimensional CommonCrawl GloVe embeddings was 1% for unigrams and 0.6% for bigrams and for 300 dimensional Amazon GloVe embeddings was 0.4% for unigrams and 0.3% for bigrams.

For the IMDB BonG vectors in table 5, we only consider unigrams and bigrams that occur at least 10 times in the train data while for SST we consider all unigrams and bigrams. We use 300 dimensional Amazon GloVe embeddings for all experiments

6 Conclusion

In this work, we have suggested the hypothesis that a lot of power of sentence/paragraph embeddings may derive from the fact that they encode the bag-of-n-grams (BonG) representation of text. Our outer product embedding (as well as its dimension-reduced version) can be seen as compressed sensing where the sensing matrix contains low-dimensional word embeddings. Using the work of Calderbank et. al. (2009)[2] showing that learning on sparse data points can be done after compressed sensing, it follows that sentence embeddings constructed even using random word vectors can approach the performance of the best SVM classifier on BonG vectors. As a corollary this shows that LSTMs are at least as powerful as the best SVM classifier on BonG vectors, which was not known before. In fact matching the performance of the best SVM classifier is a popular benchmark for the performance of sentence embeddings.

The experimental results show that the compressed sensing recovery works even better when it uses pretrained word vectors (eg word2vec or GloVe) instead of random vectors. This is unexpected given standard wisdom in compressed sensing, but does not contradict anything because the recovery only happens for vectors that correspond to actual text, and the sensing matrix (= word embeddings) was trained using actual text.

Empirically, we are able to achieve near BonG performance on sentiment analysis tasks with unigram-bigram sentence embeddings using GloVe vectors, as well as low-dimensional sketches of these embeddings. Intriguingly, sentence representations using distributional word embeddings perform as

Model Order	Embedding	Modification	dimension	IMDB	SST
Unigram	BoW	None		87.37	81.44
		SIF-weighted[1]		88.43	79.57
	GloVe	None	300	87.56	82.32
		SIF-weighted[1]	300	87.35	82.10
		Denoised	300	87.93	82.59
Bigram	BonG*	None		88.45	81.66
		SIF-weighted[1]		91.09	83.36
	GloVe†	Concat	900	87.66	81.93
		Emul	600	87.72	83.20
		OP+sample	600	87.57	82.32
		OP+JL	600	87.67	82.87
		OP+sample	3300	88.26	82.59
		OP+JL	3300	88.51	82.76
		OP	90300	89.16	83.30
Neural	Byte mLSTM[19]		4096	92.88	**91.8**
	TopicRNN[5]		300	**93.76**	-

* Concatenated with their correspondning unigram BoW.
† Concatenated with 300-dimensional unweighted unigram embedding.

Table 1: Performances on standard document classification tasks. We consider only fully unsupervised embeddings when comparing with recent neural network approaches.

d	Random Vectors			GloVe		
	Unigram	Bigram: d-sketch	Bigram: $10d$-sketch	Unigram	Bigram: d-sketch	Bigram: $10d$-sketch
100	69.13	70.64	75.50	85.12	85.64	87.07
300	75.69	76.77	80.82	87.56	87.72	88.51
1000	82.04	82.63	-	88.64	89.12	-

Table 2: Effect of dimension on performance on IMDB sentiment classification.

well or better than the random embedding representations (often believed to be the best choice for compressed sensing) leaving room for theory to explain these results in future work.

References

[1] S. Arora, Y. Liang, and T. Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. *ICLR* 2017.

[2] R. Calderbank, S. Jafarpour, and R. Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. Technical Report 2009.

[3] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the* 25^{th} *International Conference on Machine Learning* 2008.

[4] Common Crawl. 2012

[5] A. B. Dieng, C. Wang, J. Gao, and J. Paisley. TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. *ICLR* 2017.

[6] Z. Harris. Distributional Structure. *Word*, **10**(23):146-162.

[7] T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity: the Lasso and generalizations. *Monographs on Statistics and Applied Probability* **143** 2015.

[8] G. E. Hinton. Mapping Part-Whole Hierarchies into Connectionist Networks. *Artificial Intelligence* **46** 1990.

[9] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Journal of Neural Computation* **9**(8):1735-1780. 1997.

[10] M. Iyyer, V. Manjunatha, J. Boyd-Graber, H. Daumé III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. *Association for Computational Linguistics* 2015.

[11] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, **26**:189-206, 1984.

[12] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. *Proceedings of the 49th Annual ACL* 2011.

[13] J. McAuley and A. Yang. Addressing Complex and Subjective Product-Related Queries with Customer Reviews. *WWW 2016*.

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *NIPS* 2013.

[15] H. S. Paskov, R. West, J. C. Mitchell, and T. J. Hastie. Compressive Feature Learning. *NIPS* 2013.

[16] A. Pauls and D. Klein. Faster and Smaller N-Gram Language Models. *Proceedings of the 49th Annual ACL-HLT* 2011.

[17] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing* 1532-1543. 2014.

[18] T. Plate. Holographic Reduced Representations. *IEEE Transactions on Neural Networks* 1995.

[19] A. Radford, R. Josefowicz, and I. Sutskever. Learning to Generate Reviews and Discovering Sentiment. *arXiv preprint* https://arxiv.org/pdf/1704.01444.pdf. 2017.

[20] D. Sculley and C. E. Brodley. Compression and Machine Learning: A New Perspective on Feature Space Vectors. *Proceedings of the 2006 DCC* 2006.

[21] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP* 2013.

[22] S. Wang and C. D. Manning. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the 50th ACL* 2012.

[23] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards Universal Paraphrastic Sentence Embeddings. *ICLR* 2016.